REMARKS

Claims 1, 3, 5, 7, 9, 11, 13, 15, 17, 19, 21, 23, 25, 27, 29, 31, 33, 35, and 37-46 are pending in this application. Claims 1, 3, 9, 13, 15, 17, 19, 21, 23, 27, 29, 31, 33, 35, 44, 45 and 46 are amended herein. Support for the amendments to claims 1, 44, and 45 may be found in claim 13. Support for the amendments to claims 3, 9, 13, 15, 17, 19, 21, 23, 27, 29, 31, 33, and 35 may be found in the claims as filed originally. Support for the amendment to claim 46 may be found in the specification at page 23, lines 9-19. No new matter has been added. Reconsideration is requested based on the foregoing amendment and the following remarks.

Claim Rejections - 35 U.S.C. § 112:

Claims 9, 21, 33 and 35 were rejected under 35 U.S.C. § 112, second paragraph, as indefinite. The Office Action asserts in section 2, at page 2, that the recitation "said last named one processor" has no antecedent basis in claim 9. The recitation "said last named one processor" does not actually appear in claim 9. Still, claims 9, 21, 33 and 35 were amended to make them more definite, in the interest of compact prosecution only, and not for any reason of patentability. In particular, the recitation "said last-named another processor" is now "said another processor," in claims 9, 21, 33 and 35. Withdrawal of the rejection is earnestly solicited.

Claim Rejections - 35 U.S.C. § 103:

Claims 1, 3, 5, 7, 9, 11, 37-46 were rejected under 35 U.S.C. § 103(a) as being unpatentable over U.S. Patent No. 6,223,274 B1 to Catthoor et al. (hereinafter "Catthoor") in view of U.S. Patent No. 6,061,711 to Song et al. (hereinafter "Song"). The rejection is traversed to the extent it might apply to the claims as amended. Reconsideration is earnestly solicited.

Independent claims 1, 44, and 45 have been amended to include substantially the former subject matter of claim 13, which was indicated to contain allowable subject matter in section 29, at page 9 of the Office Action. Independent claims 1, 44, and 45, and the claims dependent therefrom, are thus submitted to be allowable.

In any case, in several embodiments, the storing section, the store control section, the stop control section, and the start control section of independent claims 1, 44, and 45 handle handover information. The handover information is used for performing tasks carried out on two or more processors. In particular, when the control section switches a task from one processor element to another processor element, the handover information used for the performance of the task by the first processor (i.e. before switching) is also used for the performance of the task by the second processor (i.e. after switching).

The fifth clause of claim 1 recites:

> Storing handover information relating to the common program which information is to be handed over from said one processor element to said another processor element.

Neither Catthoor nor Song teach, disclose, nor suggest "storing handover information relating to the common program which information is to be handed over from said one processor element to said another processor element," as recited in claim 1. Catthoor, rather, stores only "information relating to the status of routines running on the custom processors," as noted in the Office Action in section 5, at page 3. In particular, as described at column 30, lines 26 to 30:

> 10. The processing engine of claim 8, wherein the programmable processor has a register and the customized processor is adapted to transmit information relating to the status of routines running on the custom processor for storage in said registers.

Since Catthoor transmits information relating to the status of routines running on the custom processor for storage in the registers, Catthoor is not "storing handover information relating to the common program which information is to be handed over from said one processor element to said another processor element," as recited in claim 1.

Catthoor, furthermore, uses the information relating to the status of routines running on the custom processor to determine the *timing* of the context switch, not perform the switch itself. In particular, as described at column 30, lines 19 to 22:

> 8. The processing engine of claim 1, wherein the customized processor is adapted supply information to the programmable processor sufficient to determine the timing of the context switch.

Since Catthoor uses the information relating to the status of routines running on the custom processor to determine the timing of the context switch, Catthoor is not "storing handover information relating to the common program which information is to be handed over from said one processor element to said another processor element," as recited in claim 1.

Furthermore, in Catthoor, the *controller* switches context from the customized processor to the flexible processor at the interruption point. In particular, as described at column 29, lines 58 to 67, continuing at column 30, lines 1 and 2:

> 1. A programmable processing engine, the processing engine including a customized processor, a flexible processor and a data store commonly sharable between the two processors, the customized processor normally executing a sequence of a plurality of pre-customized routines the programmable processing

> engine, comprising:
> a controller for monitoring the customized processor during execution of a first code portion to select one of a set of pre-customized processing interruption points in a first routine and for switching context from the customized processor to the flexible processor at the interruption point.

Since, in Catthoor, the controller switches context from the customized processor to the flexible processor at the interruption point, Catthoor is not "storing handover information relating to the common program which information is to be handed over from said one processor element to said another processor element," as recited in claim 1.

Furthermore, in Catthoor, the CP 21 provides the IP 22 with *just* enough data to know how to determine the context switching point. In particular, as described at column 11, lines 44 to 52:

> In order to follow the traversal of the CP 21 through its code, the CP 21 provides the IP 22 in accordance with one embodiment of the present invention with just enough data to know how to determine the context switching point. For example, sufficient data is provided as to which of the data-dependent branches have been taken so that the exact cycle to steer the counter in the interrupt routine for the context switch can be calculated based on flags.

Since, in Catthoor, the CP 21 provides the IP 22 with just enough data to know how to determine the context switching point, Catthoor is not "storing handover information relating to the common program which information is to be handed over from said one processor element to said another processor element," as recited in claim 1.

Finally, in Catthoor, the IP 22 regularly checks on the *flags* to follow the progress of the CP 21. In particular, as described at column 11, lines 52 to 61:

> Making the flags available can be done by loading appropriate flags into registers 29 which are part of the IP register space (see FIG. 9). If such registers 29 in the IP 22 are not accessible from outside, then in principle this can alternatively happen by writes to main memory 23 but then the overhead of accessing these simple flags is much larger and the time penalty of the context switches increases significantly. The IP 22 regularly checks on these flags to follow the progress of the CP 21.

Since, in Catthoor, the IP 22 regularly checks on these flags to follow the progress of the CP 21, Catthoor is not "storing handover information relating to the common program which information is to be handed over from said one processor element to said another processor element," as recited in claim 1.

Song, for its part, is about context switching between *programs*, not processor elements. In particular, as described in Song at column 2, lines 7, 8, and 9:

> The present invention advantageously reduces the amount of processor time needed to context switch between programs.

Since Song is about context switching between programs, Song has no need for "storing handover information relating to the common program which information is to be handed over from said one processor element to said another processor element," either, and thus cannot make up for the deficiencies of Catthoor with respect to claim 1. Thus, even if Catthoor and Song were combined, the claimed invention would not result.

The third clause of claim 1 recites:

> Detecting a switching request signal to request switching such plural processor elements one from another

Catthoor neither teaches, discloses, nor suggests "detecting a switching request signal to request switching plural processor elements one from another," as acknowledged graciously in section 6 of the Office Action, at page 4. The Office Action seeks to compensate for this deficiency of Catthoor by combining Catthoor with Song.

Song, however, is about context switching between *programs*, not processor elements, as discussed above, and thus cannot make up for this deficiency of Catthoor with respect to claim 1 in any case. As described in Song, rather, at column 2, lines 52-60:

> The computing system further includes a second memory coupled to the first processor, a context switch request detector operating on the first processor for detecting, after the processor encounters one of the markers in an executing program, a request to context switch out the program, and a context saving module operating on the first processor for responding to a detected context switch request by saving in the second memory processor state information located in the first memory.

Thus, Song is about detecting a request to context switch out the program running on a first processor, and responding to the detected context switch request by saving state information located in the first memory in the second *memory* processor, not a second processor element. Therefore, even if Catthoor and Song were combined as proposed in the Office Action, the claimed invention would not result.

The Office Action seeks to justify the combination of Catthoor and Song in section 7 of the Office Action, at page 4, by saying that,

> It would have been obvious to one of ordinary skill in the art at the time the invention was made to combine the teachings of Catthoor and Song because Song's switching request signal detecting section detects the switching request signal would improve the use efficiency Catthoor's system by providing the step of switching request signal detecting section detects the switching request signal to

increase the system's performance with reducing the amount of processor state information transfer during a context switch.

Catthoor, however, has no *need* for "detecting a switching request signal to request switching plural processor elements one from another," as discussed above, since the controller switches context from the customized processor to the flexible processor at the interruption point. It is submitted, therefore, that even if Song did teach switching plural processor elements one from another, persons of ordinary skill in the art who read Catthoor for all it contained at the time the invention was made would not have been motivated to modify Catthoor as proposed in the Office Action, since it would have served no purpose. Claim 1 is submitted to be allowable. Withdrawal of the rejection of claim 1 is earnestly solicited.

Claims 3, 5, 7, 9, 11, 13, 15, 17, 19, 21, 23, 25, 27, 29, 31, 33, 35, and 37-43 depend from claim 1 and add additional distinguishing elements. Claims 3, 5, 7, 9, 11, 13, 15, 17, 19, 21, 23, 25, 27, 29, 31, 33, 35, and 37-43 are thus also submitted to be allowable. Withdrawal of the rejection of claims 3, 5, 7, 9, 11, 13, 15, 17, 19, 21, 23, 25, 27, 29, 31, 33, 35, and 37-43 is earnestly solicited.

Claim 44:

The second clause of claim 44 recites:

> Detecting a switching request signal to request switching such plural processor elements one from another.

Neither Catthoor nor Song teach, disclose, or suggest "detecting a switching request signal to request switching such plural processor elements, one from another," as discussed above with respect to claim 1

Furthermore, persons of ordinary skill in the art would have not been motivated to combine Catthoor with Song, as proposed in the Office Action, as also discussed above with respect to claim 1.

The third clause of claim 44 recites:

> Storing handover information relating to the common program, which information is to be handed over from said one processor element to said another processor element, into a storing section of said multiprocessor system.

Neither Catthoor nor Song teach, disclose, or suggest "storing handover information relating to the common program, which information is to be handed over from said one processor element to said another processor element, into a storing section of said multiprocessor system," as discussed above with respect to claim 1. Claim 44 ought thus to be allowable as well, for at least those reasons discussed above with respect to claim 1. Withdrawal of the rejection of claim 44 is earnestly solicited.

Claim 45:

The second clause of claim 45 recites:

> Detecting a switching request signal to request switching such plural processor elements one from another.

Neither Catthoor nor Song teach, disclose, or suggest "detecting a switching request signal to request switching such plural processor elements, one from another," as discussed above with respect to claim 1.

Furthermore, persons of ordinary skill in the art would have not been motivated to combine Catthoor with Song, as proposed in the Office Action, as also discussed above with respect to claim 1.

The third clause of claim 45 recites:

> Storing handover information relating to the common program, which information is to be handed over from said one processor element to said another processor element, into a storing section of said multiprocessor system.

Neither Catthoor nor Song teach, disclose, or suggest "storing handover information relating to the common program, which information is to be handed over from said one processor element to said another processor element, into a storing section of said multiprocessor system," as discussed above with respect to claim 1. Claim 45 ought thus to be allowable as well, for at least those reasons discussed above with respect to claim 1. Withdrawal of the rejection of claim 45 is earnestly solicited.

Claim 46:

The third clause of claim 46 recites:

> Storing handover information relating to the execution of the common program by said first processor element before said handover for use by said second processor after said handover.

Neither Catthoor nor Song teach, disclose, or suggest "storing handover information relating to the execution of the common program by said first processor element before said handover for use by said second processor after said handover." as discussed above with respect to claim 1.

The fourth clause of claim 46 recites:

> Wherein said first processor element and said second processor element hand over said handover information to each other through a shared memory section.

Neither Traversat nor Song teach, disclose, or suggest "said first processor element and said second processor element hand over said handover information to each other through a shared memory section,: as recited in claim 46. Claim 46 is thus believed to be allowable as well, for at least those reasons discussed above with respect to claim 1. Withdrawal of the rejection of claim 46 is earnestly solicited.

<u>Rejection of claims 1, 3, 5, 7, 9, 11, and 37-46 as unpatentable over U.S. Patent No. 6,763,440 B1 to Traversat et al. ("Traversat") in view of Song:</u>

Claims 1, 3, 5, 7, 9, 11, and 37-46 are rejected under 35 U.S.C. §103(a) as being unpatentable over U.S. Patent No. 6,763,440 B1 to Traversat <u>et al.</u> ("Traversat") in view of Song. The rejection is traversed to the extent it might apply to the claims as amended. Reconsideration is earnestly solicited.

Independent claims 1, 44, and 45 have been amended to include substantially the former subject matter of claim 13, which was indicated to contain allowable subject matter in section 29, at page 9 of the Office Action. Independent claims 1, 44, and 45, and the claims dependent therefrom, are thus submitted to be allowable.

The fifth clause of claim 1 recites:

> A storing section, responsive to each switching of said processor elements by said control section, for storing handover information relating to the common program which information is to be handed over from said one processor element to said another processor element.

Neither Traversat nor Song teach, disclose, nor suggest "a storing section, responsive to each switching of said processor elements by said control section, for storing handover information relating to the common program which information is to be handed over from said one processor element to said another processor element," as recited in claim 1. In Traversat, rather, "a check appointed state of application 104a *may* be sent to client 130 and resume executing on client 130," as noted in the Office Action in section 18, at page 6 (emphasis added). In particular, as described at column 28, lines 28 to 36:

> When migrating application 104a to client 130, a checkpointed state of application 104a may be sent to client 130. On client 130, a new virtual heap 104b may be established in persistent store 120b, a new in-memory heap 108b may be established, and application 104b may resume executing on client 130

from the checkpointed state of application 104a.

Since, in Traversat, a check appointed state of application 104a may be sent to client 130 and resume executing on client 130, the migration is discretionary, rather than "responsive to each switching of said processor elements by said control section," as recited in claim 1.

Furthermore, in Traversat, the most recently *checkpointed* persistent state of the application 104 in persistent heap 120 is packaged and sent to the client system 130. In particular, as described at column 29, lines 3 to 6:

> In step 324, the most recently checkpointed persistent state of the application 104 in persistent heap 120 is packaged and sent to the client system 130 where the application 104 is to migrate.

Since, in Traversat, the most recently checkpointed persistent state of the application 104 in persistent heap 120 is packaged and sent to the client system 130, neither the checkpointing nor the migration in Traversat is "responsive to each switching of said processor elements by said control section," as recited in claim 1.

Furthermore, in Traversat, a process's entire persistent state is copied *atomically* and committed as having migrated on both the sending and receiving client systems. In particular, as described at column 29, lines 7 to 10:

> In one embodiment, a transaction mechanism may be used, where a process's entire persistent state is copied atomically and committed as having migrated on both the sending and receiving client systems in step 328.

Since, in Traversat, a process's entire persistent state is copied atomically and committed as having migrated on both the sending and receiving client systems, Traversat is not "storing handover information relating to the common program which information is to be handed over from said one processor element to said another processor element," as recited in claim 1.

A checkpoint in Traversat, rather, is a persistent store of the state of an application and its execution environment at a *point* in time. In particular, as described at column 10, lines 40 to 44:

> At certain times, a checkpoint for application 104 may be written to persistent store space 120. In this application, a checkpoint is a persistent store of the state of an application and its execution environment (such as the virtual machine state) at a point in time.

None of the particular times in Traversat, however, are "responsive to each switching of said processor elements by said control section," as recited in claim 1.

Rather, in Traversat, a store checkpoint may be issued any time a change is made. In particular, as described at column 24, lines 44 to 47:

> In one embodiment, in order to keep the heap 108 and client system 100 states very closely synchronized with the store 120, a store checkpoint may be issued any time a change is made.

Since, in Traversat, a store checkpoint may be issued any time a change is made, the checkpointing of Traversat is not "responsive to each switching of said processor elements by said control section," as recited in claim 1.

Furthermore, in Traversat, a time may be selected to perform the checkpoint that may generate the least amount of *overhead*. In particular, as described at column 24, lines 55 to 63:

> Since the client system 100 may be interrupted to commit a store checkpoint, a time may be selected to perform the checkpoint that may generate the least amount of overhead. Fir example, when a client system performs a thread switching or synchronization, a check may be made to see if a checkpoint needs to be performed. Checkpointing may be performed asynchronously, so the overhead may be defined in terms of issuing the request, not executing the request.

Since, in Traversat, a time may be selected to perform the checkpoint that may generate the least amount of overhead, the checkpointing of Traversat is not "responsive to each switching of said processor elements by said control section," as recited in claim 1.

Furthermore, in Traversat, if many pages were updated, the changes may be committed as *soon* as possible. In particular, as described at column 24, lines 64 to 67:

> The number of pages written since the last committed checkpoint may be considered. If many pages were updated, the changes may be committed as soon as possible.

Since, in Traversat, the changes may be committed as soon as possible, the checkpointing of Traversat is not "responsive to each switching of said processor elements by said control section," as recited in claim 1.

Furthermore, in Traversat, if store operations are slow, the number of checkpoints may have to be *limited*. In particular, as described at column 25, lines 1 to 5:

> The speed at which a transaction checkpoint can be performed to the store may be considered. If store operations are slow, the number of checkpoints may have to be limited. Buffering may not be an option in a tight memory environment.

Since, in Traversat, the number of checkpoints may have to be limited, the checkpointing of Traversat is not "responsive to each switching of said processor elements by said control section," as recited in claim 1.

Furthermore, in Traversat, a checkpoint may be performed after a *garbage* collection. In particular, as described at column 25, lines 6, 7, and 8:

> A checkpoint may be performed after a garbage collection, since many pages may have changed due to object collections and heap compacting.

Since, in Traversat, a checkpoint may be performed after a garbage collection, the checkpointing of Traversat is not "responsive to each switching of said processor elements by said control section," as recited in claim 1.

Finally, in Traversat, a checkpoint may be induced after a *maximum* elapsed time. In particular, as described at column 25, lines 12, 13, and 14:

> To save the computation state, a checkpoint may be induced after a maximum elapsed time, if none of the previous conditions occurred.

Since, in Traversat, a checkpoint may be induced after a maximum elapsed time, the checkpointing of Traversat is not "responsive to each switching of said processor elements by said control section," as recited in claim 1.

Song, for its part, is about context switching between *programs*, not processor elements, as discussed above. Since Song is about context switching between programs, Song has no need for "storing handover information relating to the common program which information is to be handed over from said one processor element to said another processor element," either, and thus cannot make up for the deficiencies of Traversat with respect to claim 1. Thus, even if Traversat and Song were combined, the claimed invention would not result.

The third clause of claim 1 recites:

> Detecting a switching request signal to request switching such plural processor elements one from another

Traversat neither teaches, discloses, nor suggests "detecting a switching request signal to request switching plural processor elements one from another," as acknowledged graciously in section 19 of the Office Action, at page 7. The Office Action seeks to compensate for this deficiency of Traversat by combining Traversat with Song.

Song, however, is about context switching between *programs*, not processor elements, as discussed above, and thus cannot make up for this deficiency of Traversat with respect to claim 1 in any case. Therefore, even if Traversat and Song were combined as proposed in the Office Action, the claimed invention would not result.

The Office Action seeks to justify the combination of Traversat and Song in section 20 of the Office Action, at pages 7 and 8, by asserting that,

> It would have been obvious to one of ordinary skill in the art at the time the invention was made to combine the teachings of Traversat and Song because Song's switching request signal detecting section detects the switching request signal would improve the use efficiency Traversat's system by providing the step of switching request signal detecting section detects the switching request signal to increase the system's performance with reducing the amount of processor state information transfer during a context switch.

In Traversat, however, a *checkpointed* state of application 104a is migrated to client 130 from persistent store 120, as discussed above, not the application 104a executing on client 100. Traversat, therefore, has no need for "detecting a switching request signal to request switching plural processor elements one from another," since no "switching plural processor elements one from another," is even taking place.

It is submitted, therefore, that even if Song did teach switching plural processor elements one from another, persons of ordinary skill in the art who read Traversat for all it contained at the time the invention was made would not have been motivated to modify Traversat as proposed in the Office Action, since it would have served no purpose. Claim 1 is submitted to be allowable. Withdrawal of the rejection of claim 1 is earnestly solicited.

Claims 3, 5, 7, 9, 11, 13, 15, 17, 19, 21, 23, 25, 27, 29, 31, 33, 35, and 37-43 depend from claim 1 and add additional distinguishing elements. Claims 3, 5, 7, 9, 11, 13, 15, 17, 19, 21, 23, 25, 27, 29, 31, 33, 35, and 37-43 are thus also submitted to be allowable. Withdrawal of the rejection of claims 3, 5, 7, 9, 11, 13, 15, 17, 19, 21, 23, 25, 27, 29, 31, 33, 35, and 37-43 is earnestly solicited.

Claim 44:

The second clause of claim 44 recites:

> Detecting a switching request signal to request switching such plural processor elements one from another.

Neither Traversat nor Song teach, disclose, or suggest "detecting a switching request signal to request switching such plural processor elements, one from another," as discussed above with respect to claim 1

Furthermore, persons of ordinary skill in the art would have not been motivated to combine Traversat with Song, as proposed in the Office Action, as also discussed above with respect to claim 1.

The third clause of claim 44 recites:

> In response to each switching of said processor elements, storing handover information relating to the common program, which information is to be handed over from said one processor element to said another processor element, into a storing section of said multiprocessor system.

Neither Traversat nor Song teach, disclose, or suggest "in response to each switching of said processor elements, storing handover information relating to the common program, which information is to be handed over from said one processor element to said another processor element, into a storing section of said multiprocessor system," as discussed above with respect to claim 1. Claim 44 ought thus to be allowable as well, for at least those reasons discussed above with respect to claim 1. Withdrawal of the rejection of claim 44 is earnestly solicited.

<u>Claim 45:</u>

The second clause of claim 45 recites:

> Detecting a switching request signal to request switching such plural processor elements one from another.

Neither Traversat nor Song teach, disclose, or suggest "detecting a switching request signal to request switching such plural processor elements, one from another," as discussed above with respect to claim 1.

Furthermore, persons of ordinary skill in the art would have not been motivated to combine Traversat with Song, as proposed in the Office Action, as also discussed above with respect to claim 1.

The third clause of claim 45 recites:

> In response to each switching of said processor elements, storing handover information relating to the common program, which is to be handed over from said one processor element to said another processor element into a storing section of said multiprocessor system.

Neither Traversat nor Song teach, disclose, or suggest "in response to each switching of said processor elements, storing handover information relating to the common program, which is to be handed over from said one processor element to said another processor element into a storing section of said multiprocessor system," as discussed above with respect to claim 1. Claim 45 ought thus to be allowable as well, for at least those reasons discussed above with respect to claim 1. Withdrawal of the rejection of claim 45 is earnestly solicited.

Claim 46:

The third clause of claim 46 recites:

> Responsive to said switching of said processor elements by said control section, for storing handover information relating to the execution of the common program by said first processor element before said handover for use by said second processor after said handover.

Neither Traversat nor Song teach, disclose, or suggest "responsive to said switching of said processor elements by said control section, for storing handover information relating to the execution of the common program by said first processor element before said handover for use by said second processor after said handover," as discussed above with respect to claim 1.

The fourth clause of claim 46 recites:

> Wherein said first processor element and said second processor element hand over said handover information to each other through a shared memory section.

Neither Traversat nor Song teach, disclose, or suggest "said first processor element and said second processor element hand over said handover information to each other through a shared memory section, as recited in claim 46. Traversat, rather, appears to checkpoint the application into the persistent store, to wait there until such time as the application *might* be migrated to another system. Claim 46 is thus believed to be allowable as well, for at least those reasons discussed above with respect to claim 1. Withdrawal of the rejection of claim 46 is earnestly solicited.

Allowable Subject Matter:

The Applicant acknowledges with appreciation the indication that claims 13, 19, 21, 25, 29, 31, 33, and 35 contain allowable subject matter.

Conclusion:

Accordingly, in view of the reasons given above, it is submitted that all of claims 1, 3, 5, 7, 9, 11, 13, 15, 17, 19, 21, 23, 25, 27, 29, 31, 33, 35, and 37-46 are allowable over the cited references.

Finally, if there are any formal matters remaining after this response, the Examiner is requested to telephone the undersigned to attend to these matters.

If there are any additional fees associated with filing of this Amendment, please charge the same to our Deposit Account No. 19-3935.

Respectfully submitted,

STAAS & HALSEY LLP

Date: 11 AP 07

By: [illegible]
Thomas E. McKiernan
Registration No. 37,889

1201 New York Avenue, NW, 7th Floor
Washington, D.C. 20005
Telephone: (202) 434-1500
Facsimile: (202) 434-1501